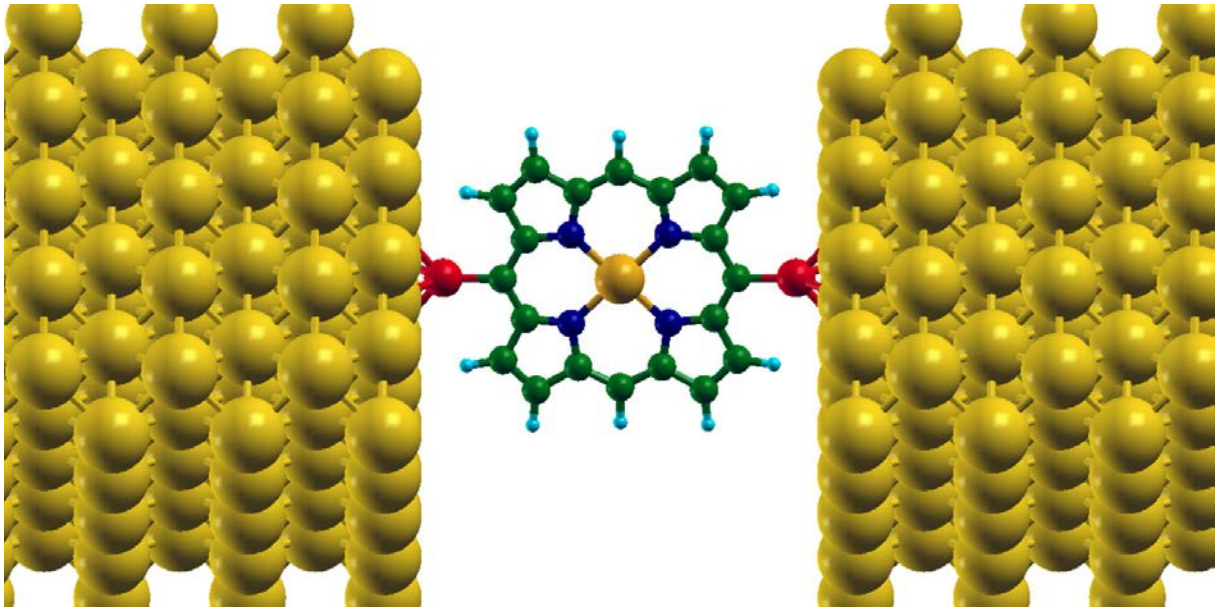
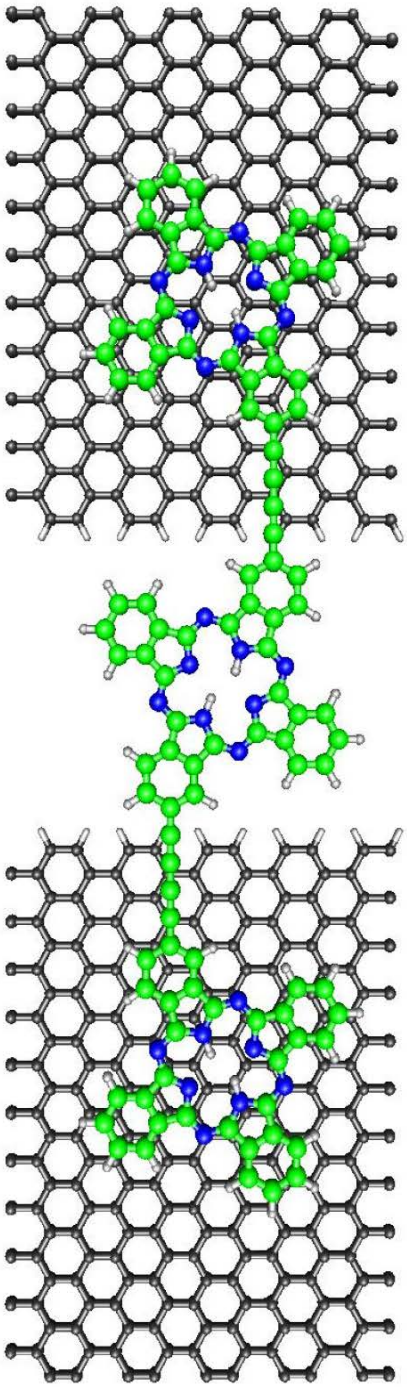
A next-generation simulation tool for electron, thermal and spin transport.

GOLLUM 1.0

User Manual

June 2014





GOLLUM Authors

Steven Bailey	**Lancaster University**
Jaime Ferrer	**Universidad de Oviedo**
Victor García-Suárez	**Universidad de Oviedo**
Colin J. Lambert	**Lancaster University**
Hatef Sadeghi	**Lancaster University**

GOLLUM Team

Laith Algharagholy	**Lancaster University**
Iain Grace	**Lancaster University**
Kaitlin Guillemot	**Lancaster University**
David Z. Manrique	**Lancaster University**
Laszlo Oroszlani	**Eötovos University**
Rubén Rodríguez-Ferradás	**Universidad de Oviedo**
David Visontai	**Lancaster University**

1 CONTENTS

2 INTRODUCTION

2.1 UPDATES AND NEW FUNCTIONALITY.

Updates to bug fixes together with code development will be posted on the web-site:
http://www.physics.lancs.ac.uk/gollum
Currently none of the GOLLUM versions is explicitly parallelized.

2.2 THE ADVISORY BOARD:

All changes will be vetted by the advisory board:
Jaime Ferrer ferrer@uniovi.es
Colin J. Lambert c.lambert@lancaster.ac.uk
Victor García-Suárez vm.garcia@cinn.es
Hatef Sadeghi h.sadeghi@lancaster.ac.uk
Steven Bailey s.bailey@lancaster.ac.uk
Please report bugs to the GOLLUM mailing list: http://www.physics.lancs.ac.uk/gollum/mail

2.3 CITING GOLLUM

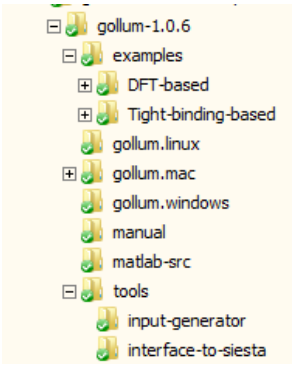
A full description of the functionality and capabilities of the code are to be found in:
The New Journal of Physics : J Ferrer et al 2014 New J. Phys. **16** 093029.
doi:10.1088/1367-2630/16/9/093029

2.4 GOLLUM UNITS

1. Energy (E) in eV.
2. Current intensities (I) in Amps.
3. Voltages (V) in volts.
4. Temperatures in Kelvins.
5. All energies are referred to the Fermi energy (E_F) of the EM region.

2.5 DISTRIBUTED FOLDER STRUCTURE

The distribution contains the following folders



Depending upon your operating system you will find in folders 'gollum.linux, gollum.mac and gollum.windows' the executable file plus a generated Readme.

The 'examples' folder is more complex but contains all the information to run the tutorial examples. Once the '.zip' files are extracted you will find two folders 'DFT-based' and 'Tight-binding-based' each of which will expand into the MACOSX versions and the Windows/Linux versions.

All the 'Tight-binding-based' examples labelled 1-6 contain:

- 'mode-1, mode-2, mode-3, mode-4' output data folders.
- 'Guide.pdf' which gives details of the system and plots of the expected calculation result.
- 'input' showing the GOLLUM input file.
- 'Extended_Molecule' input to GOLLUM
- 'Lead_1' input to GOLLUM
- 'Lead_2' input to GOLLUM

All the 'DFT-based' examples labelled 1-8 contain:

- 'mode-1, mode-2, mode-3, mode-4' output data folders.
- Guide.pdf' which gives details of the system and plots of the expected calculation result.
- 'input' showing the GOLLUM input file.
- The input files needed to run the associated SIESTA [2] calculations are to be found in 'lead.fdf' and 'emol.fdf'. We also include the pseudopotential files ('.psf'). NOTE the ',fdf' files have the flags needed to extract the Hamiltonians using the DIRECT SIESTA INTERFACE METHOD shown in section 6.1.
- 'input' showing the GOLLUM input file.
- 'Gollum_files' with the 'Extended_Molecule, Lead_1 and Lead_2' GOLLUM input files.

The folder 'manual' contains this documentation.

The folder 'tools' contains the 'indirect-siesta-interface' to construct GOLLUM compatible Hamiltonians using the 'gollum-input-generator' with a handy GOLLUM input file generator in MATLAB and 'direct-siesta-interface' with the Fortran routines to install into the SIESTA package as an option to generate the GOLLUM compatible Hamiltonians.

For developers only the MATLAB source code is found in 'matlab-src'.

3 THE SCOPE OF THE GOLLUM PROJECT.

GOLLUM is a program that computes the electrical and thermal transport properties of multi- terminal nanoscale scale systems. The program can compute transport properties of either user-defined systems described by a tight-binding (or Hückel) Hamiltonian, or more material-specific properties of systems composed of real atoms described by DFT Hamiltonians. The program has been designed to interface easily with any DFT code, which uses a localized basis set and currently read information from all the latest public flavors of the codes SIESTA [2] (SIESTA 3.1, SIESTA VDW, SIESTA LDA+U) and FIREBALL [3]. Plans to generate interfaces to other codes are underway. GOLLUM is based on equilibrium transport theory, which means that it consumes much less memory than non-equilibrium Green's function codes. The program has been designed for user-friendliness and takes a considerable leap towards the realization of ab initio multi-scale simulations of conventional and more sophisticated transport functionalities. These include:

- The ability to use either model tight-binding or DFT Hamiltonians, including the optional use of scissor corrections.
- The ability to compute non-equilibrium current-voltage curves.
- Access to the full scattering matrix, including scattering amplitudes, phases and Wigner delay times.
- An ability to construct conductance statistics relevant to break-junction and STM measurements of single-molecule conductances.
- Integration of a wide range of phenomena, including thermoelectrics, spintronics, superconductivity, Coulomb blockade, quantum pumps, Kondo physics and topological phases.
- The ability to describe multi-probe structures.
- An interface between classical molecular dynamics, which handles interactions with the environment.

The GOLLUM work-flow shown in FIG. 3.1 below illustrates the versatility of the package to allow the user to generate the required Hamiltonians either 'by hand' from prescribed tight-binding parameters or by adapting the output of DFT codes. A suitable methodology follows the following route. Once the atomic arrangements are generated these are fed into the second stage, where the Hamiltonian matrix is generated. This stage is in practice independent of the previous geometry construction and can be run separately, taking only the output geometries of the first stage. The junction Hamiltonian can be generated using a variety of tools, some of which are listed in box II in FIG 3.1. A popular approach is the use of DFT codes that are able to write the Hamiltonian in a tight-binding language. In this way, model tight-binding Hamiltonians can also be easily generated. Other approaches involve the use of Slater-Koster or semi-empirical methods. In addition, GOLLUM has the ability to modify these Hamiltonian matrices as discussed above. For example, the Hamiltonian matrix can be modified to include scissor corrections, Coulomb-blockade physics, a gate or bias voltage, a magnetic phase factor or a superconducting order parameter. Finally, stage III is the actual quantum transport calculation. This takes the Hamiltonian matrix as an input and calculates the s-matrix and associated physical quantities, such the electrical or spin current, the conductance or the thermopower.

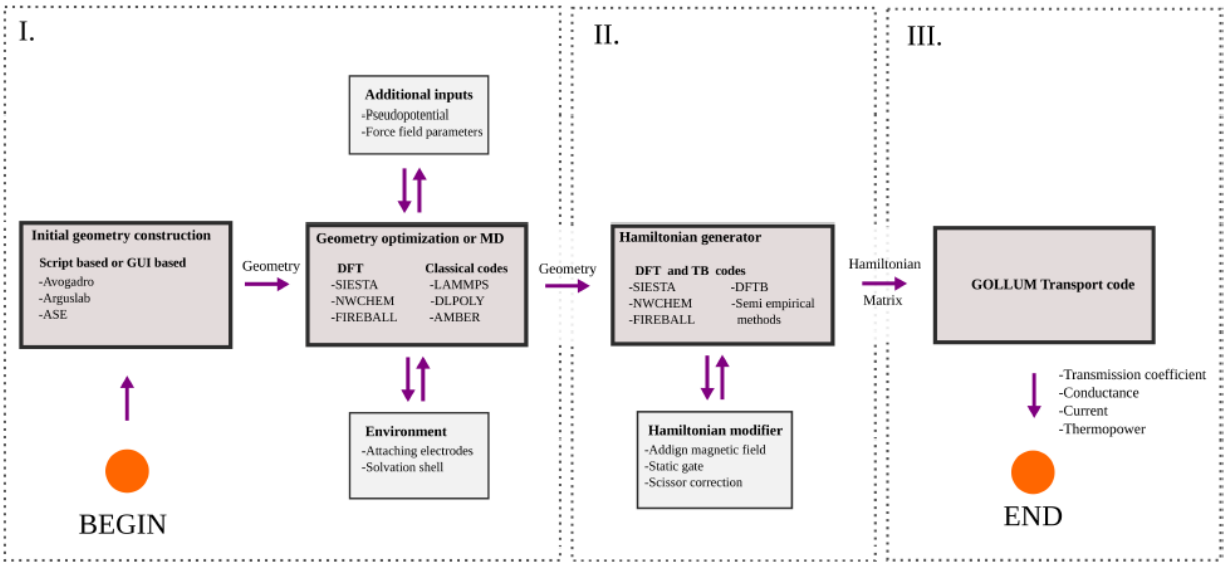


FIG.3.1: Typical GOLLUM workflow with various optional software tools.

4 INSTALLATION AND RUNNING.

GOLLUM is distributed as a standalone MATLAB executable. A copy can be obtained by logging onto the web page at http://www.physics.lancs.ac.uk/gollum/ and completing the application form.

Upon request a user specific compatible version of GOLLUM will be provided for example to run on 32 bit machines.

Requests for the developer-version will be considered by the advisory board.

4.1 MATLAB VERSION.

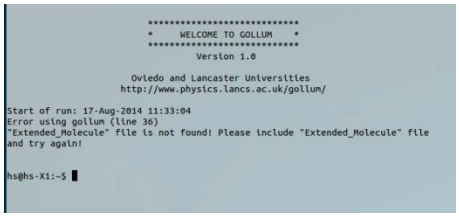
As GOLLUM is a standalone executable, all you need to run the programme is a GOLLUM- compatible MATLAB Compiler Runtime (MCR).

4.1.1 INSTALLING THE MCR AND RUNNING IN WINDOWS

From your GOLLUM distribution navigate to your version dependent executable in the 'gollum.windows' directory to locate 'gollum.exe'.

For a windows machine, to verify if you have the MCR installed simply: double click on gollum.exe or run gollum.exe from the command prompt.

If you have a compatible MCR, then you will see the following window:



Ignore the 'Error' at this stage as it indicates that as yet you have not included the correct input files to run a calculation.

If you do not have the correct MCR then navigate to http://www.mathworks.co.uk/products/compiler/mcr/ in MathWorks and locate the correct version to install.

> Windows requires release Windows R2012a 64-bit. (If you requested the 32 bit version of GOLLUM, then you will need the release Windows R2012a 32-bit.)

Once the MCR is correctly installed you can make your first test.

You can of course place gollum.exe in your path (For example in Windows 7 follow: Control Panel, System and Security, System, Advanced System Settings, Environment variables, then scroll down system variables to find path, click on path then edit) and use the command prompt to run the code but for simplicity we outline the following procedure.

Copy from the examples folder supplied with your distribution an example of 'your choice' into a new directory of your choice. Place 'gollum.exe' in the folder and run the calculation by double clicking

'gollum.exe'. Your results of the calculation should match those to be found in the example matching 'your choice'.

4.1.2 INSTALLING THE MCR AND RUNNING LINUX

From your GOLLUM distribution navigate to your version dependent executable in the 'gollum.linux' directory to locate 'gollum' and 'run_gollum.sh'.

For a LINUX machine verify that you have the MCR e.g. using the terminal command

locate MATLAB_Compiler_Runtime

If you do not have the correct MCR then navigate to http://www.mathworks.co.uk/products/compiler/mcr/ in MathWorks and locate the correct version to install.

> LINUX requires Linux R2012a 64-bit. (If you requested the 32 bit version of GOLLUM, then you will need the release Linux R2012a 32-bit.)

Install as root to place the application in your 'usr/local directory'.

Enter your 'gollum_linux' directory to find 'gollum' and 'run_gollum.sh' NOTE: ensure that you have full permissions in place for both files.

To test your MCR locate your MCR libraries which should be in
/usr/local/MATLAB/MATLAB_Compiler_Runtime/v717/

Then type

./run_gollum.sh /usr/local/MATLAB/MATLAB_Compiler_Runtime/v717/

A successful compile is seen by the following window:

Ignore the 'Error' at this stage as it indicates that as yet you have not included the correct input files to run a calculation.

The location of the MCR libraries and 'gollum' can be placed in your PATH by editing your .bashrc or .cshrc files or by typing the following to set the path prior to each calculation.
ln -s path-source/run_gollum.sh run_gollum.sh
ln -s path-source/gollum Gollum

For simplicity we outline the following procedure.
Copy from the examples folder supplied with your distribution an example of 'your choice' into a new directory of your choice. Place 'gollum' and 'run_gollum.sh' into the folder.
Type

./run_gollum.sh /usr/local/MATLAB/MATLAB_Compiler_Runtime/v717/

Your results of the calculation should match those to be found in the example matching 'your choice'.

4.1.3 INSTALLING THE MCR AND RUNNING MAC

Enter the 'gollum.mac' directory to locate the folder 'gollum.app' which contains the executable and the files 'readme.txt' and 'run_gollum.sh'

locate MATLAB_Compiler_Runtime

If you do not have the correct MCR then navigate to http://www.mathworks.co.uk/products/compiler/mcr/ in MathWorks and locate the correct version to install.

Mac requires release Mac R2014a intel 64bit.

Once the MCR is correctly installed you can make your first test. Follow the instructions in the MATLAB readme file making sure that you place not only the 'run_gollum.sh' but the 'gollum.app' in your test directory and type: ./run_gollum.sh <mcr_directory> where mcr_directory locates the MCR version you have installed.

If you have a compatible MCR, then you will see the following window:

Ignore the 'Error' at this stage as it indicates that as yet you have not included the correct input files to run a calculation.

You can of course place 'gollum'in your path using the Mac terminal in a similar way to the Linux procedure to run the code.

To run an example for simplicity we outline the following procedure.
Copy from the examples folder supplied with your distribution an example of your choice into a new directory.
Place 'gollum.app' and 'run_gollum.sh' into the folder.
Type

./run_gollum.sh <mcr_directory>

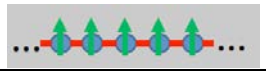
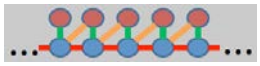
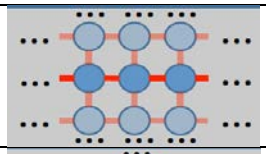
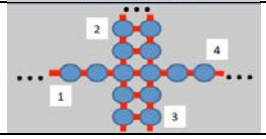
Your results of the calculation should match those to be found in the example.

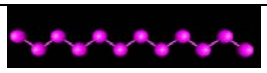
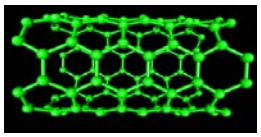
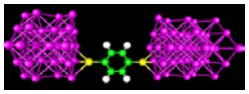
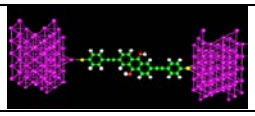
5 TEST EXAMPLES AND TUTORIALS.

The tutorials are to be found as a set of calculations in 'gollum/examples' in your distribution. There are 'Tight-binding-based' (6 examples) for MAC and Windows/Linux MATLAB and 'DFT-based' (8 examples) for MAC and Windows/Linux MATLAB. The Example files and related systems are summarised in TABLE 5.1. We will take the first and simplest example from both the Tight-binding and DFT-based sets and run through the steps to carry out the calculation and relate the steps to the details found in this manual. The more advanced cases are then clearly explained from the 'Guide.pdf' accompanying each of the examples.

It is important to run each of the examples to ensure that you are happy with your compilation by then comparing the results to be found in the output data 'mode' folders and the plots shown in the 'Guide'pdf'. Following all the examples and reading this document together with the GOLLUM [1] paper will complete the tutorial.

5.1 SUMMARY OF THE EXAMPLES: TABLE 5.1.

	Guide-based information.	
Tight-binding-based example	**System**	**Comment**
1.Single-atom_chain.Spin		Simulate a spin-polarized one-dimensional (1D) system with perfect transmission equal to the number of open channels or bands at a given energy.
2.Single-atom_chain.Non-coll		Simulate a one-dimensional (1D) system with perfect transmission equal to the number of open channels or bands at a certain energy and non-collinear magnetic configuration.
3.Two-level_system.SAINT		Simulate a two level system (H2 molecule) coupled to metallic atomic chains. Check that the transmission has two peaks corresponding to the positions of the levels of the molecule. Apply a gate voltage to change the on site energies and the SAINT correction to increase the separation between both molecular levels.
4.Two-level_chain		Simulate a one-dimensional (1D) system with different orbitals and perfect transmission equal to the number of open channels or bands at a given energy.
5.Square_lattice		Simulate a two-dimensional (2D) system with perfect transmission equal to the number of open channels or bands at a given energy.
6.Four-probe_system		Simulate a four-probe system with two types of electrodes. Calculate the transmissions from one to other electrodes. See the effect of including a bias voltage on one of the electrodes. Calculate the currents.
DFT-based example	**System**	**Comment**
1.Au_linear_chain		Simulate a one dimensional (1D) system with perfect transmission equal to the number of open channels or bands at a certain energy.
2.Ir_zigzag_chain		Simulate a one dimensional (1D) system with perfect transmission equal to the number of open channels or

		bands at a certain energy, various channels and a non collinear magnetic configuration.
3.C_chain+gap.NDR		Simulate a one-dimensional (1D) system with a vacuum gap to see the effect of including a bias voltage that falls in the middle.
4.Ni_001_bulk		Simulate a magnetic three-dimensional (3D) system (transverse periodic boundary conditions and *k*-points) with perfect transmission equal to the number of open channels or bands at a certain energy and various channels.
5.CNT_5,5		Simulate a quasi-one-dimensional (q1D) system with perfect transmission equal to the number of open channels or bands at a certain energy and various channels.
6.Graphene		Simulate a perfect two-dimensional (2D) system transverse periodic boundary conditions and *k*-points along *y* with perfect transmission equal to the number of open channels or bands at a certain energy and various channels.
7.Au_001+BDT.SAINT		Calculate the transport properties of a BDT (1,4-benzenedithiolate) molecule between gold electrodes. Correct the transport properties by applying a spectral adjustment (SAINT) to the occupied and unoccupied levels and shifting the Fermi level towards the HOMO.
8.Au_111+OPE		Calculate the transport properties of a OPE (oligophenyleneethylene) based molecule between gold electrodes.

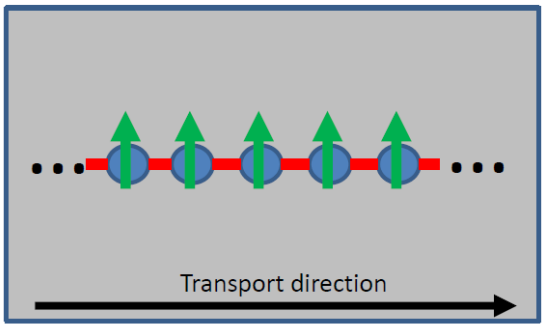
To run these examples simply follow the same procedure set out in section 4 INSTALLATION AND RUNNING above.

5.2 STEP BY STEP EXAMPLE: TIGHT-BINDING-BASED.

5.2.1 1. SINGLE-ATOM_CHAIN.SPIN

Enter the directory: \gollum-1.0\examples\Tight-binding-based\1.Single-atom_chain.Spin from your distribution.

Single-atom chain. Spin polarized



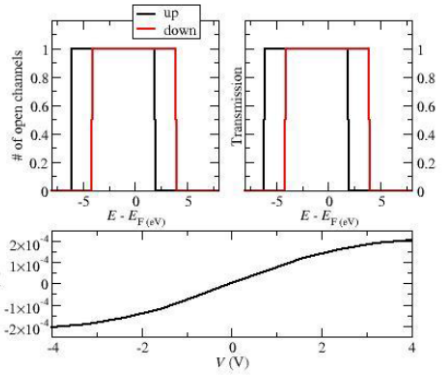
Objectives

- Simulate a spin-polarized one-dimensional (1D) system with perfect transmission (equal to the number of open channels or bands at a certain energy).
- Check that the transmission exactly coincides with the number of open channels.

Results

- The transmission has a step-like shape which corresponds to the number of open channels.
- The spin-up and spin-down channels and transmissions have an exchange splitting of 2 eV.
- The current linearly increases (as expected, since it is equal to the integral of a constant function) until it saturates at large voltages (when the bias window starts covering the edges of the band, which are "rounded" by the effect of the voltage).

Transmission and number of open channels as a function of energy. Calculated with **Mode 1** of Gollum.



Current as a function of voltage. Calculated with **Mode 4** of Gollum.

System description and parameters

Leads

- Unit cell with just one atom. One orbital per atom.
- Single-atom chain with spin polarization. The exchange splitting between the spin-up and spin-down states is 2 eV.

Extended molecule (EM)

- 5 atoms in the calculation.
- Same parameters as in the leads.

Gollum parameters

- Transmission coefficients calculated between -8.0 eV and 8.0 eV in 200 energy points.
- 2 principal layers on each electrode. The terminating principal layer is the second on each. Electronic structure obtained from the leads calculation.
- No SAINT correction or Fermi level shift.
- Bias voltage between -4.0 V and 4.0 V calculated in 11 voltage points. The bias shift is applied on the first terminating layer of the left and right electrodes only (4th column of the `atom` variable set to different from 0). This is an approximation since the voltage should not fall (or at least fall very slowly) on a perfect infinite system.

Tight-binding parameters

- Orthogonal basis set (overlap matrix = identity matrix).
- On-site energies (both leads and EM): -1.0 eV (spin-up) and 1.0 eV (spin-down).
- Nearest-neighbour coupling matrix elements (both leads and EM): -2.0 eV.

Transfer the files: 'input, Lead_ 1, Lead_2 and Extended_Molecule' to a new test folder of your choice. Execute GOLLUM as described in SECTION 4 INSTALLATION and RUNNING and check your results.

The Hamiltonians are generated by hand from these tight binding parameters and are shown in 'Leads_1 and 2' with the 'Extended_Molecule'.

The 'input' file is defined in SECTION 8.1.

The 'Extended_Molecule' file is defined in SECTION 8.2

The 'Lead-1 and Lead_2' files is defined in SECTION 8.3

The output data files are given for the various 'mode' directories and are described in SECTION 9.

5.3 STEP BY STEP EXAMPLE: DFT-BASED.

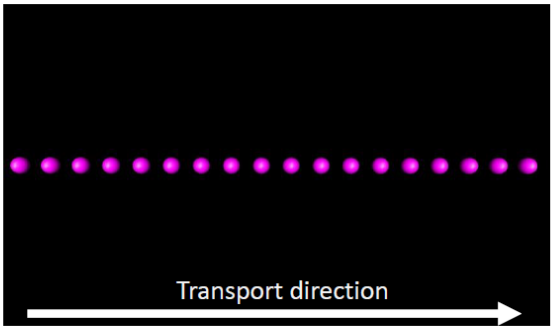
5.3.1 1.AU_LINEAR_CHAIN

Enter the directory: \gollum-1.0\examples\DFT-based\1.Au_linear_chain.

Gold linear chain

Objectives

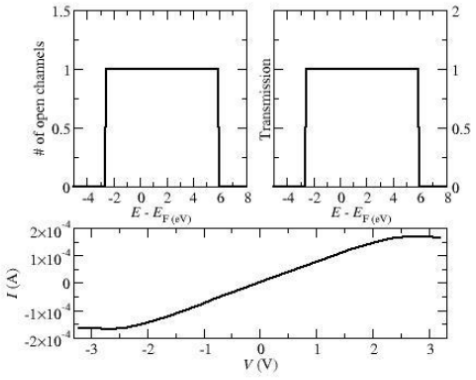
- Simulate a one-dimensional (1D) system with perfect transmission (equal to the number of open channels or bands at a certain energy).
- Check that the transmission exactly coincides with the number of open channels.



Results

- The transmission has a step-like shape which corresponds to the number of open channels. In this case there is only one s orbital in the valence of gold (the rest of states are included in the pseudopotential), so the transmission is equal to 1 in the range of energies that comprises the gold band.
- The current linearly increases (as expected, since it is equal to the integral of a constant function) until it saturates and slightly decreases (when the bias window starts covering the edges of the band, which are "rounded" by the effect of the voltage).

Transmission and number of open channels as a function of energy. Calculated with **Mode 1** of Gollum.



Current as a function of voltage. Calculated with **Mode 4** of Gollum.

System description and parameters

Leads

- Au leads made of only 2 gold atoms.
- Lattice vectors long enough along the transverse directions to make sure the system is 1D.

Extended molecule (EM)

- Perfect gold chain made of 18 atoms (9 unit cells of the electrodes) and with the same transverse lattice vectors of the leads.
- The system is also periodic along the transport direction (z) to avoid finite size effects.

Gollum parameters

- Transmission coefficients calculated between -5.0 eV and 8.0 eV in 200 energy points.
- 2 principal layers on each electrode. The terminating principal layer is the second on each. Electronic structure obtained from the leads calculation.
- No SAINT correction or Fermi level shift.
- Bias voltage between -3.2 V and 3.2 V calculated in 25 voltage points. The bias shift is applied on the first terminating layer of the left and right electrodes only (4th column of the `atom` variable set to different from 0). This is an approximation since the voltage should not fall (or at least fall very slowly) on a perfect infinite system.

Ab-initio (Siesta) parameters

- Basis set: Single-ζ (SZ); only the s orbital is included in the Au valence (the rest are hidden in the pseudopotential).
- LDA exchange and correlation functional. CA parametrization.
- Mesh cut-off: 150 Ry.
- 90 k-points along z in the lead calculation. 10 k-points in the EM calculation (to make sure the electronic structure exactly matches that of the leads). 1 k-point (Γ point) along the perpendicular directions.

Files necessary tor run the **leads** calculation: lead.fdf, Au.psf
Files necessary to run the **EM** calculation: emol.fdf, Au.psf

In this case the definitions of the Hamiltonians are generated from ab-initio DFT parameters using SIESTA and are shown in 'Gollum_files\Leads_1 and 2' with the 'Extended_Molecule'. These files are generated using the method explained in SECTION 6 HAMILTONIAN GENERATION and in SECTION 6.2 DIRECT SIESTA INTERFACE METHOD. This makes use of the files 'lead.fdf and emol.fdf' with 'Au.psf' provided within the distribution. You will need to obtain a licenced SIESTA version (http://departments.icmab.es/leem/siesta/) and understand how to run the code from the documentation.

Again transfer the files: 'input, Lead_ 1, Lead_2 and Extended_Molecule' to a new test folder. Execute GOLLUM as described in SECTION 4 INSTALLATION and RUNNING and check your results.

The 'input' file is defined in SECTION 8.1.

The 'Extended_Molecule' file is defined in SECTION 8.2

The 'Lead-1 and Lead_2' files is defined in SECTION 8.3

The output data files are given for the various 'mode' directories and are described in SECTION 9.

6 HAMILTONIAN GENERATION.

The Tight Binding Hamiltonians can be constructed by hand to generate the 'Lead_i' and 'Extended_Molecule' input files used by GOLLUM, nevertheless we strongly recommend that the Hamiltonians are generated through a suitable DFT code. At present GOLLUM is directly interfaced with SIESTA and the various directly compatible versions can be found in 'direct-siesta-interface'. Nevertheless GOLLUM is not restricted to these SIESTA versions and a suitable Hamiltonian can be obtained from other versions by using the INDIRECT SIESTA INTERFACE METHOD shown in SECTION 6.2 below.

6.1 DIRECT SIESTA INTERFACE METHOD.

The direct interface requires that SIESTA is recompiled with the GOLLUM interface subroutines. In the folder 'direct-siesta-interface' the files that need to be included in the SIESTA main source folder are located. Using the included Makefile will then compile a GOLLUM-compatible SIESTA executable. Running SIESTA will then produce the 'Extended_Molecule' and or the 'Lead_i' files if the flag EMol or Lead is included in the SIESTA '.fdf' input file. Examples of the flag usage are to be found in 'examples/DFT-based/DFT-based/'. Other interfaces with flavours of SIESTA are also included:

Interface-siesta-3.0-rc2

Interface-siesta-LDA+U

Interface-siesta-spinorbit

Interface-siesta-trunk-367-VdW

6.2 INDIRECT SIESTA INTERFACE METHOD

GOLLUM is designed to be flexible. It can use any suitable DFT Hamiltonian by constructing the GOLLUM-compatible Hamiltonian. At present we have developed a set of routines to convert SIESTA output files as follows.

6.2.1 FILES NEEDED FROM THE SIESTA CALCULATION.

In order to generate the files that are used by Gollum to calculate the transport properties it is necessary first to redirect the Siesta output to a file:

A. Some of the information needed by Gollum is already printed in the output file of Siesta (the file where the Siesta output is redirected), e.g. Ni.out

It is also necessary to include some options in the Siesta 'Ni.fdf' file, so that Siesta prints the necessary information: firstly name the output files accordingly and

B. "SaveHS T". With this option in the 'Ni.fdf' file Siesta generates a '.HSX' file e.g. Ni.HSX where the Hamiltonian and overlap matrix elements are stored.
C. "WriteEigenvalues T". With this option in the '.fdf' file, Siesta generates a '.EIG' e.g. Ni.EIG file where the Fermi energy and the eigenvalues are stored.

Siesta should also generate by default a KP file e.g. Ni.KP where the *k*-points information is stored.

6.2.2 HAMILTONIAN INTERFACE FILES AND STRUCTURE.

The interface is included in the directory 'tools/ interface-to-siesta'. There are two Fortran files (.f90) and a 'Makefile'

A. hsx_m.f90: Module that reads and processes the Hamiltonian and overlap from the .HSX file.
B. siesta2gollum.f90: Main program, which reads the rest of the information and creates the Extended_Molecule or Lead_i files.
C. Makefile: Makes the *siesta2gollum* executable. Edit the 'Makefile' to specify the compiler after the FC (Fortran compiler) variable, e.g. for a *ifort* compiler: "FC = ifort"

6.2.3 GENERATION OF THE GOLLUM FILES.

Once the *siesta2gollum* executable is generated, the files that are needed to run GOLLUM should be produced by specifying the .out file and whether the calculation was based on a lead or extended molecule calculation (notice that the *siesta2gollum* file should be either in the same directory as the rest of the Siesta files or linked to that directory):

A. Leads: *siesta2gollum* "Siesta output file" "Integer value between 1 and 123456789 that represents the electrodes where the lead files are going to be used".
   i. For example, if a calculation of a nickel lead has been done (Siesta output dumped to e.g. the Ni.out file) and it is going to be used in two electrodes of a certain system, do the following: "./siesta2gollum Ni.out 12" . This generates the files Lead_1 and Lead_2, which should be included in the directory where Gollum is going to be run.

   ii. Another example: If a gold lead simulation has been performed (Siesta output dumped to e.g. Au_lead.out file) and such lead is going to be used in electrodes 2, 7 and 9 of a multiterminal calculation, do the following: "./siesta2gollum Au_lead.out 279" . This generates the files Lead_2, Lead_7 and Lead_9, which should be included in the directory where Gollum is going to be run.

6.2.4 EXTENDED MOLECULE

A. siesta2gollum "Siesta output file" "0".
i. For example, if a simulation of the extended molecule of an iridium cluster between silver surfaces has been performed (Siesta calculation dumped to e.g. IrAg.out) do the following: "./siesta2gollum IrAg.out 0". This generates the Extended_Molecule file that should be included in the directory where Gollum is going to be run.

6.3 EXTRAS 1.

In 'tools/input-generator' there is a MATLAB script to help generate the INPUT lists and convert the format of the INPUT files for GOLLUM. For large simulations the input files are tedious to write out by hand and the MATLAB script input-generator.m reads input-generator.in to automate the process.

7 THEORETICAL APPROACH

For a full description please refer to the GOLLUM [1] paper.

7.1 NOMENCLATURE.

GOLLUM describes open systems comprising an extended scattering region (coloured dark blue in FIGS. 7.1.1 and 7.1.2) connected to external crystalline leads (coloured light blue in FIGS. 7.1.1 and 7.1.2). Depending on the problem of interest and the language used to describe the system, the material (M) of interest forming the central part of the scattering region could comprise a single molecule, a quantum dot, a mesoscopic cavity, a carbon nanotube, a two-dimensional mono or multi-layered material, a magneto-resistive element or a region containing one or more superconductors. FIG. 7.1.1 shows an example of a 4-lead system whose central scattering region (labeled M) is a molecule. It is important to note that in an accurate ab initio description of such a structure, the properties of the leads closest to the molecule (or more generally the central scattering material) will be modified by the presence of the central scattering region (M) and by the fact that the leads terminate. In what follows, we refer to those affected portions of the leads closest to the central scatterer as 'branches' and include them as part of the 'extended scatterer' (denoted EM). Consequently within GOLLUM, a typical structure consists of an extended scatterer (EM), formed from both the central scatterer (M) and the branches. The extended scattering region is connected to crystalline current-carrying leads of constant cross-section, shown in light blue in the FIGS. 7.1.1 and 7.1.2. For an accurate description of a given system, the branches are chosen to be long enough such that they join smoothly with the (light blue) crystalline leads. Crucially, the properties of this interface region between the central scatterer M and the leads are determined by their mutual interaction and are not properties of either M or the electrodes alone.

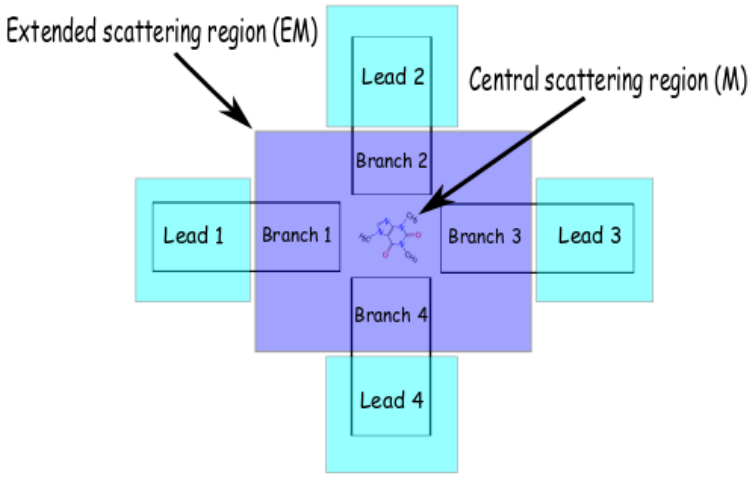


FIG. 7.1.1: Schematic plot of a four-terminal device, which includes an extended scattering region and four leads.

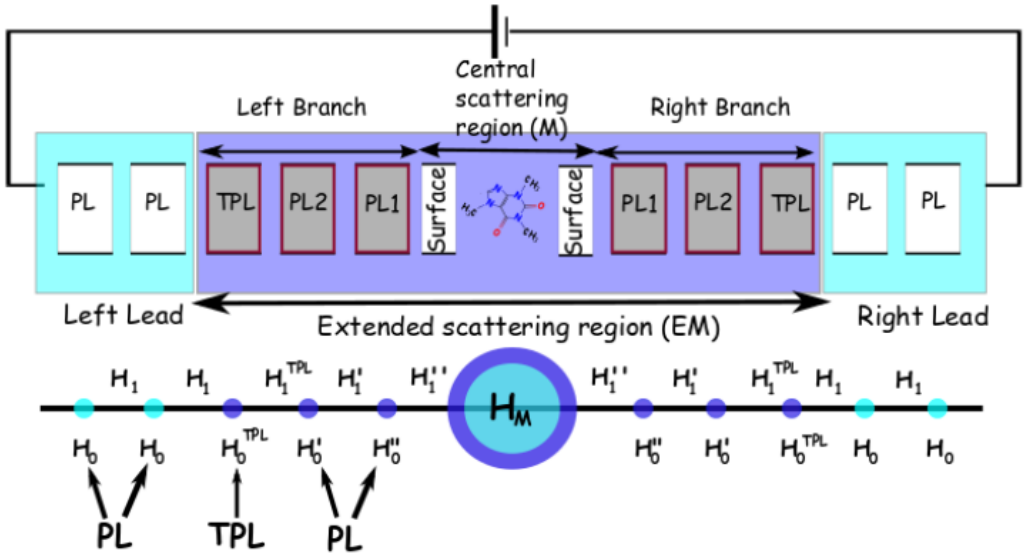


FIG. 7.1.2: (Top) Schematic two-terminal device, where electrons are driven from the left to the right lead through the extended scattering region. The leads are possibly kept at chemical potentials $\mu_{1,2} = -eV_{1,2}$, where $V = V_1 - V_2$ is the applied bias. (Bottom) Each lead is composed of an infinite chain of PLs with Hamiltonians H_0, which are coupled with each other via the coupling Hamiltonians H_1. The Extended Molecule comprises the actual Molecule, the electrodes surfaces and the Left and Right Branches. These branches contain several PLs up to the TPL. The TPL link the EM to the leads. Both leads are assumed to be identical in this figure for simplicity.

FIG. 7.1.2 shows a two-terminal device in more detail and introduces further terminology. The regions in light blue are called electrodes or leads and are described by perfect periodic Hamiltonians subject to chosen chemical potentials. Each lead i is formed by a semi-infinite series of identical layers of constant cross section, which we refer to as principal layers (PLs). FIG. 7.1.1 shows only two PLs per lead (coloured white), although an infinite number is implied. Furthermore in the figure, the leads are identical and therefore the lead index i has been dropped. These PLs are described mathematically by intra-layer Hamiltonians H_0^i. PLs must be chosen so that they are coupled only to their nearest neighbors by the Hamiltonians H_1^i. If each PL contains N^i orbitals then H_0^i and H_1^i are square $N^i - N^i$ matrices.

The extended scatterer (EM) in dark blue is composed of a central scattering region (M) and branches. Each branch contains several PLs. These PLs have an identical atomic arrangement as the PLs in the leads. However, their Hamiltonians differ from H_0^i and H_1^i due to the presence of the central scattering region. PL numbering at each branch starts at the PL beside the central scattering region. The outermost PL at each branch of the EM region is called the terminating principal layer (TPL) and must be described by Hamiltonians $H_0^{i,TPL}$ and $H_1^{i,TPL}$ which are close enough to H_0^i and H_1^i, to match smoothly with the corresponding lead Hamiltonian. For this reason, GOLLUM requires that the EM contain at the very least one PL. The central scatterer (M) itself is described by an intra-scatterer Hamiltonian H_M^0 and coupling matrices to the closest PLs of the branches. In the example in FIG. 7.1.1, the central scattering region M comprises a molecule and the atoms forming the electrode surfaces. The surfaces in GOLLUM include all atoms belonging to the electrodes whose atomic arrangements cannot be cast exactly as a PL, due to surface reconstructions, etc. For simplicity, FIG. 7.1.1 shows the case of a symmetric system, although no such symmetries are imposed by GOLLUM. All Hamiltonians are spin- dependent, but again for notational simplicity, the spin index σ is not written explicitly.

7.2 THE SURFACE GREEN'S FUNCTION FOR THE CURRENT CARRYING LEADS.

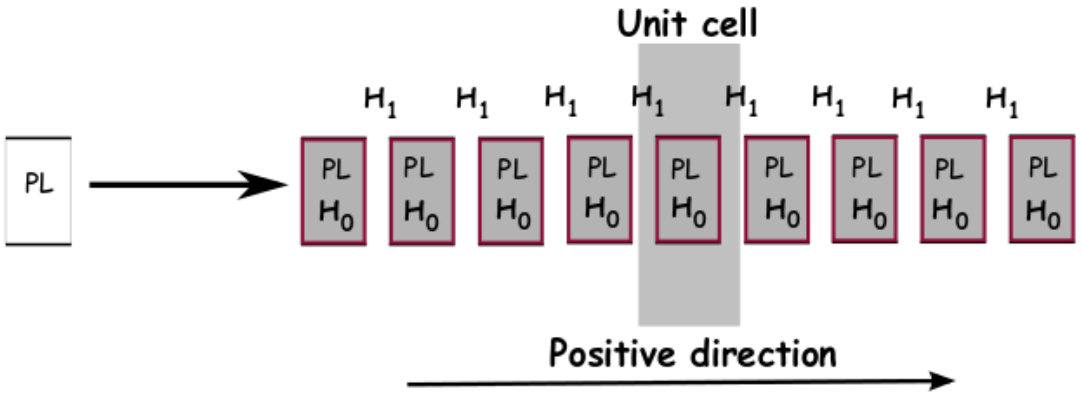


FIG.7.2.1 Illustrates the infinite system used to generate the leads Hamiltonians K_0^i and K_1^i where the positive direction is towards the scatterer. For non-orthogonal basis sets the overlap matrices ($S_{o,\pm1}^i$) and (S^i) must have the same structure as the Hamiltonian matrices and

$$K_{o,\pm1}^i = H_{o,\pm1-}^i ES_{o,\pm1}^i \quad (1)$$

$$K^i = H^i - ES^i \quad (2)$$

By expanding the Bloch eigenstates $|\Psi(k)\rangle = \sum_{n,\mu} e^{ikn}\, c_\mu(k)|\varphi(n,\mu)\rangle$ of the infinite system in a localised basis set a NxN secular equation can be found $(K_0 + K_1 e^{ik} + K_{-1}e^{-ik})C(k) = 0$ which if by choosing the energy (E) and solving for the allowed wave vectors produces $2N$ solutions with either real or complex wave vectors $(k_p\ p = 1 \ldots..2N)$ which GOLLUM solves to give the eigenvalues and eigenvectors $C(k_p)$.

$$\begin{pmatrix} -K_0 & -K_{-1} \\ I_N & 0_N \end{pmatrix} C(k_p) = e^{ik_p} \begin{pmatrix} K_1 & 0_N \\ 0_N & I_N \end{pmatrix} C(k_p) \quad (3)$$

The group velocities of the states corresponding to the real wave vectors is then given by

$$v(k_p) = i \frac{C(k_p)^\dagger \left(K_1 e^{ik_p} - K_{-1} e^{-ik_p}\right) C(k_p)}{C(k_p)^\dagger \left(S_0 + S_1 e^{ik_p} + S_{-1} e^{-ik_p}\right) C(k_p)} \quad (3a)$$

$v(k_p)$ has units of energy and is real and the fully-dimensioned group velocity is given by $v(k_p)(a/h)$ where a is the spacing between PL's in a given lead. The $2N$ wave vectors are split into two sets the first denoted k_p are real (complex) wave vectors that have $v_p > 0$ (Im $k_p > 0$) and therefore propagate (decay) towards the right of FIG. 7.2. These are called positive open (closed) channels. The second set are denoted $\tilde{k}_p$ are real (complex) wave vectors that have $v_p < 0$ (Im $k_p < 0$) and therefore propagate (decay) towards the left of FIG. 5. These are called negative open (closed) channels. By using the dual vectors $D(k_p)$ and $D(\tilde{k}_p)$ which satisfy $D(k_p)^\dagger . C(k_q) = \delta_{p,q}$ and $D(\tilde{k}_p)^\dagger . C(\tilde{k}_q) = \delta_{p,q}$ which are found by inverting the NxN matrices:

$$Q = \left(C(k_1), \ldots\ldots, C(k_N)\right) = (C_1, \ldots\ldots C_N)$$

$$\tilde{Q} = \left(C(\tilde{k}_1), \ldots\ldots, C(\tilde{k}_N)\right) = (\tilde{C}_1, \ldots\ldots \tilde{C}_N)$$

$$(D_1, \ldots \ldots D_N) = (Q^{-1})^{\dagger}$$

$$\left(\widetilde{D}_1, \ldots \ldots \widetilde{D}_N\right) = (\tilde{Q}^{-1})^{\dagger}$$

the required transfer matrices are constructed:

$$T = \sum_1^N C_n e^{i k_n} D_n^{\dagger} \quad (3b)$$

$$\tilde{T} = \sum_1^N \tilde{C}_n e^{i \tilde{k}_n} \widetilde{D}_n^{\dagger} \quad (3c)$$

The transfer matrices allow us to build the coupling matrix V, the self-energies Σ and the surface Green's functions $G_{S,0}^i$:

$$V = K_{-1}(T^{-1} - \tilde{T}) \quad (4)$$

$$\Sigma = K_1 T \quad (5)$$

$$G_{S,0}^i = -(K_0 + \Sigma)^{-1} \quad (6)$$

NOTE the procedure might fail if K_1 is singular. We adopt a decimation method to remove the offending degrees of freedom (see the GOLLUM paper for details [1]).

7.3 THE EXTENDED SCATTERING REGION HAMILTONIAN

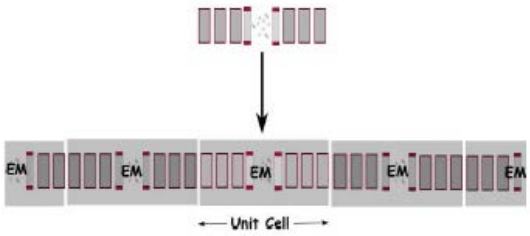


FIG.7.3.1. The infinite system where the unit cell is linked by periodic boundary condition.

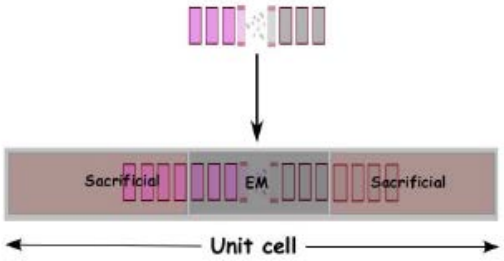


FIG.7.3.2. The super-cell containing the EM with vacuum buffer regions to left and right.

The extended scattering region Hamiltonian K^{EM} can be provided as a model Hamiltonian, or generated by a DFT or other material-specific program. One of the strengths of GOLLUM is an ability to treat interfaces with high accuracy. In a tight-binding description, tight-binding parameters of a particular material are often chosen by fitting to a band structure. However this does not solve the problem of choosing parameters to describe the interface between two materials. Often this problem is finessed by choosing interface parameters to be a combination of pure material parameters such as an arithmetic or geometric mean, but there is no fundamental justification for such approximations. Therefore we describe here methods to generate K^{EM} using a DFT program, where the inclusion of branches as part of the extended scatterer occurs naturally. FIG.7.3.1 shows an example of a junction where the electrodes are identical. The system is composed of super-cells formed from a central scatterer and PLs. There are periodic boundary conditions in the longitudinal direction, such that the TPL of one branch of a super-cell is linked smoothly to the TPL of a neighboring super-cell. Running a DFT program for such a super-cell then automatically generates K^{EM}. Provided the super-cells

contain sufficient PLs, the Hamiltonians $K_0^{TPL}$and K_1^{TPL} associated with the TPLs will be almost identical to those generated from a calculation involving an infinite periodic lead, i.e. if the Hamiltonians K_0andK_1 associated with the PLs are generated from a calculation involving an infinite periodic lead, then provided the super-cells contain sufficient PLs, these will be almost identical to K_0^{TPL} and K_1^{TPL}respectively. In this case then there will be minimal scattering caused by the junction between the TPL and the lead. Clearly there is a trade-off between accuracy and CPU time, because inserting more PLs increases the size and cost of the calculation. In practice, the number of PLs retained in such a super-cell is increased in stages until the results do not change significantly as the number of PLs is increased further. There exist situations where the electrodes are dissimilar, either chemically, or because of their different crystalline structure, or because their magnetic moments are not aligned. In these cases, there cannot be a smooth matching between TPLs of neighboring super-cells in FIG.7.3.1. To address this situation, we use a setup similar to that displayed in FIG.7.3.2, where additional PLs are appended to the branches in the EM region. These additional PLs are terminated by artificial surfaces and surrounded by vacuum. The TPLs are then chosen to be one of the PLs near the middle of each branch and should be surrounded by enough PLs both towards its artificial surface and towards the central scattering region. Then the PLs placed between the TPL and the artificial surface can be discarded. These sacrificial PLs ensure that the chosen TPL is unaffected by the presence of the artificial vacuum boundary. Clearly, calculations of this sort are more expensive in numerical terms than those performed with super-cells generated as in FIG.7.3.1, because they contain many more atoms.

7.4 HAMILTONIAN ASSEMBLY

In an ab initio calculation of the transport properties of a junction, the DFT program produces the Hamiltonians and Fermi energy of the EM and leads in separate runs as the Hartree potential is defined up to a constant, which is usually different for the EM and for each lead. This usually means that the energy of the EM and of the corresponding lead PLs, as well as their Fermi energies do not agree with each other, so Eq. (1) must be rewritten as follows:

$$K_{o,\pm1}^{i} = H_{o,\pm1}^{i} - (E_F^i + E)S_{o,\pm1}^{i} \tag{7}$$

where we have referred the energy of each lead to its own Fermi energy. To fix the Hamiltonian mismatch we define a realignment variable for each lead as follows:

$$\Delta^i = H_0^i(\mu,\mu) - H_0^{EM}(\mu,\mu) \tag{8}$$

where μ indicates a relevant orbital or group of orbitals. Then, the Hamiltonian of each lead is realigned with that of the EM

$$\begin{aligned} K_{o,\pm1}^{i} &= H_{o,\pm1}^{i} - \left(E_F^{EM} + E + \Delta^i\right)S_{o,\pm1}^{i} \\ &= H_{o,\pm1}^{i} - (E_F^i + E)S_{o,\pm1}^{i} \end{aligned} \tag{9}$$

It turns out that the renormalized Fermi energies of each lead do not match perfectly with each other if the number of PLs in the EM region is not sufficiently large. This is the case when for efficiency reasons, it is desirable to artificially minimize the size of the EM. Sometimes it is advisable to choose the Fermi energy of one of the leads as the reference energy. In this case the overall shift is given by Δ^1 or the quantity $\delta = E_F^1 - E_F^{EM}$.

7.5 SCATTERING MATRIX AND TRANSMISSION IN MULTI-TERMINAL DEVICES.

We note that the most general scattering state in a given lead i at a given energy E can be written as a linear combination of open and closed channels as follows

$$|\Phi^i(E)\rangle = \sum_{k_i} o_{k_i} \frac{|\Psi^i(k_i)\rangle}{\sqrt{v_{k_i}}} + \sum_{\bar{q}_i} o_{\bar{q}_i} \frac{|\Psi^i(\bar{q}_i)\rangle}{\sqrt{v_{\bar{q}_i}}} + |\chi^i\rangle \quad (10)$$

where k_i and $\bar{q}_i$ denote here open positive and negative channels and $|\Psi^i(k_i)\rangle$ and $|\Psi^i(\bar{q}_i)\rangle$ are their normalized kets. Here, the contribution of all the closed channels in lead i is described by the ket $|\chi^i\rangle$. Consequently, the number of electrons per unit time flowing between two adjacent PLs within the lead is $j^i(E) = \sum_{k_{i\in i}} |o_{k_i}|^2 - \sum_{\bar{q}_{i\in i}} |o_{\bar{q}_i}|^2$ (11)

We pick in this section the convention that positive direction in the lead means flow towards the EM region and vice versa. So positive (negative) open channels are also called incoming (outgoing) channels of lead i. With this notation the wave-function coefficients of the incoming open channels of a given lead are determined by the properties of the reservoir connected to the lead. The wave function coefficients of the open outgoing channels of lead i are obtained from the amplitudes of all incoming channels by

$$o^i_{\bar{q}_i} = \sum_{j,k} s^{ij}_{\bar{q}_i k_j} o^j_{k_j} \quad (12)$$

Where $\bar{q}_i(k_j)$ is an outgoing (incoming) dimensionless wave-vector of lead i or lead j. It is therefore convenient to assemble the wave functions of the M^i incoming (outgoing) open channels of a given lead i in the column vectors $\bar{O}^i, O^i$ and all the scattering matrix elements connecting leads i and j into the matrix block S^{ij} of dimension $M^i \times M^j$. Therefore the above equation can be written

$$\begin{pmatrix} \bar{O}^1 \\ \bar{O}^2 \\ \cdots \\ \bar{O}^P \end{pmatrix} = \begin{pmatrix} S^{11} & S^{12} & \cdots & S^{1P} \\ S^{21} & S^{22} & \cdots & S^{2P} \\ . & . & . & . \\ S^{P1} & S^{P2} & \cdots & S^{PP} \end{pmatrix} \begin{pmatrix} O^1 \\ O^2 \\ \cdots \\ O^P \end{pmatrix} \quad (13)$$

By normalizing the Bloch eigenvectors and their duals to unit flux by

$$C^i(k_i) = \frac{C^i(k_i)}{\sqrt{v}k_i}, D^i(k_i) = \sqrt{v_{k_i}}(k_i)$$
$$C^i(\bar{q}_i) = \frac{C^i(\bar{q}_i)}{\sqrt{v}\bar{q}_i}, D^i(\bar{q}_i) = \sqrt{v_{\bar{q}_i}}(\bar{q}_i) \quad (14)$$

The matrix elements of the scattering matrix block connecting leads i and j can then be written as

$$s^{ij}_{\bar{q}_i k_j} = D^i(\bar{q}_i)\left(G_S^{ij} V^i - I\delta_{ij}\right) C^i(k_i) \quad (15)$$

Here G_S^{ij} is the off diagonal block of the surface Greens function defined in Eq. 6 that connects leads i and j and V^i is also defined in Eq.6.
With the above if the incoming channel k_i of lead i is occupied with probability $f_{k_i}(E)$ (i.e. in Eq.10 $o_{k_i} = 1$ with probability $f_{k_i}(E)$), then the number of electrons per unit time enetreing the scattering region from reservoir i along channel k_i with energy between E and $E + dE$ is

$$dI^{out}_{\bar{q}_i}(E) = (dE/h)\, f_{k_i}(E) \quad (16)$$

And the number per unit time per unit energy leaving the scatterer and entering reservoir i along channel $\bar{q}_i$ with energy between E and $E + dE$ is

$$dI^{out}_{\bar{q}_i}(E) = (dE/h)\sum_{j,k}\left|s^{ij}_{\bar{q}_i k_j}\right|^2 f_{k_i}(E) \tag{17}$$

In many cases, the incoming and outgoing channels of each lead i can be grouped into channels possessing particular attributes (i.e. quantum numbers) labelled α_i, β_i … … … … etc. This occurs when all incoming channels of a particular type α_i in lead i possess the same occupation probability $f^i_\alpha(E)$. For example, all quasi-particles of type α_i in reservoir i may possess a common chemical potential μ_{α_i} and $f^i_{\alpha_i}(E)$ may take the form $f^i_{\alpha_i}(E) = f(E - \mu_{\alpha_i})$, where $f(E)$ is the Fermi function. In this case, if the incoming and outgoing channels of type α_i belonging to lead i possess wave-vectors $k_{\alpha_i}, \bar{q}_{\alpha_i}$, then the number of quasi-particles per unit time of type α_i leaving reservoir i with energy between E and $E + dE$ is

$$dI^i_{\alpha_i}(E) = (dE/h)\sum_{j,\beta_j} P^{i,j}_{\alpha_i\beta_j} f^j_{\beta_j}(E) \tag{18}$$

where $P^{i,j}_{\alpha_i\beta_j} = M^i_{\alpha_i}(E)\delta_{i,j}\delta_{\alpha_i\beta_j} - \sum_{\bar{q}_{\alpha_i}k_{\beta_j}}\left|s^{ij}_{\bar{q}_{\alpha_i}k_{\beta_j}}\right|^2$ (19)

and $M^i_{\alpha_i}(E)$ is the number of open incoming channels of type α energy E in lead i. Note that in the above summation $\bar{q}_{\alpha_i}$ runs over all outgoing wave vectors of energy E and type α_i of lead i and k_{β_j} runs over all incoming wave vectors of energy E and type β_j in lead j. If i and j are different leads then $s_{\bar{q}_i k_j}$ is often called the transmission amplitude and denoted $t_{\bar{q}_i k_j}$ whilst if they are of the same lead then $s_{\bar{q}_i k_i}$ is called the reflection amplitude $r_{\bar{q}_i k_i}$. Similarily if $i \neq j$ it is common to define the transmission coefficient as

$$T^{i,j}_{\alpha_i\beta_j} = \sum_{\bar{q}_{\alpha_i}k_{\beta_j}}\left|s^{ij}_{\bar{q}_{\alpha_i}k_{\beta_j}}\right|^2 \tag{20}$$

For $i = j$ we define the reflection coefficient as

$$R^{i,i}_{\alpha_i\beta_i} = \sum_{\bar{q}_{\alpha_i}k_{\beta_i}}\left|s^{ij}_{\bar{q}_{\alpha_i}k_{\beta_i}}\right|^2 \tag{21}$$

So that

$$dI^i_{\alpha_i}(E) = (dE/h)\left\{\left[\sum_{\beta_i} M^i_{\alpha_i}(E)\delta_{\alpha_i\beta_i} - R^{i,i}_{\alpha_i\beta_i}\right]f^i_{\beta_i}(E) - \sum_{j\neq i,\beta_j} T^{i,j}_{\alpha_i\beta_j} f^j_{\beta_j}(E)\right\} \tag{22}$$

Note the unitarity of the scattering matrix requires that

$$\sum_{i,\bar{q}_{\alpha_i},\alpha_i}\left|s^{ij}_{\bar{q}_{\alpha_i}k_{\beta_j}}\right|^2 = \sum_{j,k_{\beta_j},\beta_j}\left|s^{ij}_{\bar{q}_{\alpha_i}k_{\beta_j}}\right|^2 = 1 \tag{23}$$

Hence the sum of the elements of each row and column of matrix P is zero:

$$\sum_{j,\beta_j} P^{i,j}_{\alpha_i\beta_j} = \sum_{i,\alpha_i} P^{i,j}_{\alpha_i\beta_j} = 0 \tag{24}$$

Or equivalently,

$$\sum_{\beta_i} R^{i,i}_{\alpha_i\beta_i} + \sum_{j\neq i,\beta_j} T^{i,j}_{\alpha_i\beta_j} = M^i_{\alpha_i} \tag{25}$$

and

$$\sum_{\alpha_i} R^{j,j}_{\alpha_i\beta_j} + \sum_{j\neq i,\alpha_i} T^{i,j}_{\alpha_i\beta_j} = M^{j}_{\beta_j} \quad (26)$$

From Eqs. 23 and 24 if $f^{j}_{\beta_j}(E)$ is independent of j and β_j then $dI^{i}_{\alpha_i}(E) = 0$ for all i and α_i. For this reason in the above equations $f^{j}_{\beta_j}(E)$ can be replaced by $\bar{f}^{j}_{\beta_j}(E) = f^{j}_{\beta_j}(E) - f(E)$ where $f(E)$ is an arbitrary function of energy which in practice is usually chosen to be the Fermi function evaluated at a convenient reference temperature and chemical potential. When comparing theory with experiment we are usually interested in computing the flux of some quantity Q from a particular reservoir. From Eq. 18 if the amount of Q carried by quasi particles of type α_i is $Q_{\alpha_i}(E)$ then the flux of Q from reservoir i is

$$I^{i}_{Q} = \int (dE/h) \sum_{\alpha_{i,j},\beta_j} Q_{\alpha_i}(E)\, P^{i,j}_{\alpha_i\beta_j} \bar{f}^{j}_{\beta_j}(E) \quad (27)$$

In the simplest case of a normal conductor choosing $Q_{\alpha_i} = -e$ independent of α_i then the above equation yields the electrical current from lead i. Within GOLLUM α_i may represent spin and in the presence of superconductivity may represent hole (h) or particle (p) degrees of freedom. In the latter case the charge $Q_p = -e$ and $Q_h = +e$.

7.5.1 GOLLUM DELIVERS THE FOLLOWING FUNCTIONALITIES:

1. Simulates multi-terminal devices
2. Reads either Tight-Binding or DFT Hamiltonians
3. Computes the full scattering matrix
4. Computes charge transport
    1. Number of open scattering channels
    2. Transmission and reflection coefficents
    3. Shot-noise
5. Computes heat transport
    1. Thermal conductance
    2. Thermopower (Seebeck coefficent)
    3. Peltier coefficent
6. Computes spin transport
    1. Collinear spins systems
    2. Non-collinear spins systems
    3. Systems with strong spin-orbit interaction
7. Computes zero-voltage as well as I-V curves
8. Handles vdW and LDA+U functionals
9. Scissors corrections scheme for strongly correlated systems
10. Computes band-structure of the leads and density of states (DOS) in the scattering region
11. Covers large samples enabling to analise ballistic to diffusive regime

Version 2.0 will come soon with

- ** Kondo & Coulomb blockade physics
- ** Magnetic field and integer quantum Hall effect
- ** Info about Local charge, spin, current and spin-current densities inside the scattering region
- ** Scripts to handle multiscale simulations

8 INPUT FILES FORMAT AND NOTATION.

Gollum can read the Hamiltonian of the extended scattering region (which includes a terminal principle layer as shown in FIG.4) and extract the PL to build the leads Green's function. It can also read stand-alone lead Hamiltonians to obtain the same information. Therefore GOLLUM requires as a minimum an input file and an Extended_Molecule file with an optional instruction to read Lead_i files.The variables in the input file are designed to be clear and easy to interpret and apart from the Mode variable which must appear first, the variables can be placed in arbitrary order and some variables are optional. The Extended_Molecule file instructs GOLLUM how to read the Hamiltonian, overlap matrix, orbitals, spin degrees of freedom and the number of transverse k points for the extended scattering region. All variables are mandatory.

An example of an input variable:

```
# name:   Bias
# type:    matrix
# rows:   4
# columns:  3
2  3.2  50
0   0   0
0   0   0
0   0   0
```

The variable name is Bias and it is a matrix containing four rows and three columns with the last four lines assigning values to the variables.

8.1 THE INPUT FILE.

8.1.1 MANDATORY VARIABLE: MODE

As you have completed the Examples to test your compilation you will note that the results appear in results/Mode-j where j=1…4. The concept of a Mode is used in GOLLUM to define the range of calculations to be carried out by GOLLUM. The basic calculation carried out by GOLLUM appears in Mode-1. Historically this was the first set of calculations that the transport code could perform and is the default value if Mode is not given a defining number.

```
# name:   Mode
#type:    scalar j
```

Mode-1 [j = 1]

In this mode, GOLLUM computes the equilibrium transport properties of a junction.

Mode-2 [j = 2]

In this mode, GOLLUM computes the zero-voltage, zero-energy spin-dependent Scattering matrix and Transmission coefficients.

Mode-3 [j = 3]

In this mode, GOLLUM computes the zero-voltage thermal properties of the junction as a function of temperature.

Mode-4 [j = 4]

In this mode, GOLLUM determines the finite-voltage transmission coefficients and current-voltage $I - V$ curves of multi-lead junctions.

8.1.2 A SUMMARY OF THE MODE FUNCTION: TABLE 9.1.2.

Before giving the details of the next input variables and their inclusion for each Mode the organisation of the input structure is summarised below.

	A	B	C	D	E	F	G	H	I	J	K
Mode-1	✔	✔	✔	✔				✔	✔	✔	✔
Mode-2	✔	✔	✔					✔	✔	✔	✔
Mode-3	✔	✔	✔		✔			✔	✔	✔	✔
Mode-4	✔	✔	✔			✔	✔	✔	✔	✔	✔

KEY	MV	PMV	OV	Definition
A: Mode	✔			Mode
B: Leadp	✔			Lead(s) PL and TPL properties
C: atom	✔			EM atom properties
D: ERange		✔		Energy range and number of data points
E: TRange		✔		Temperature range and number of data points
F: Bias		✔		Current-voltage range and number of data points
G: Biasaccuracy			✔	Energy grid refinement
H: EFshift			✔	Fermi energy shift
I: scissor			✔	Scissor corrections
J: anderson			✔	Coulomb blockade and Kondo physics
K: Vgate			✔	EM gate voltage

MV(✔) = Mandatory variable: PMV(✔) = Partial mandatory variable: OV(✔)= Optional variable
NOTE: Mode-3 functions with two leads only.

8.1.3 MANDATORY VARIABLE: LEADP

This variable contains one row per branch (or lead) and sets the branch properties. The format is:
# name: leadp
# type: matrix
# rows: 2
# columns: 3
leadp(1) leadp(2) leadp(3)

For each row (branch), the columns are as follows:

leadp(1)Set the number of PL in the branch.

leadp(2)Set the terminating PL in the branch. Notice that leadp(2)must be equal to or smaller than leadp(1).

leadp(3)This variable sets the Hamiltonian used to compute the Lead Green's function (GF). Possible values are:

0. The lead GF is computed using the Hamiltonian and overlap matrices of the TPL, using the Hamiltonian of the EM which is read from the file extended scattering region.

±**1**. The Hamiltonian and overlap matrices of ideal leads are appended at the end of the TPL and become the new TPL. These are read from the file Lead_i, where i is the Lead number. The Lead GF is computed using this new TPL. The sign indicates whether the lead incoming channels are directed along the positive (−1) or negative (+1) z-axis.

As an example, for a junction containing two leads, lead 1 (left lead) should use leadp(3)=-1 and lead 2 (right lead) should use leadp(3)=1.

For DFT-based calculations, it is advisable to generate and use the files Lead1 and Lead2, setting leadp(3)=±1.

8.1.4 MANDATORY VARIABLE: ATOM

This variable defines the properties of all the atoms in the extended scattering region. There is therefore a row per atom. For example, for an EM region containing 26 atoms, the variable would look like
# name: atom
# type: matrix
# rows: 26
# columns: 4
atom(1) atom(2) atom(3) atom(4)

For each row, the columns are defined as follows:

atom(1): Atom number in the DFT simulation.

atom(2): Sets the EM region where the atom is placed. Possible regions are "Molecule + Surface" (atom(2)=0) or one of the branches. E.g.: For Lead i, atom(2)= i.

atom(3): If atom(2) = *i* /= 0, then this variable sets the PL in Lead *i* to which the atom belongs to. However, if the atom belongs to the "Surface + Molecule" region (atom(2)=0), then this variable is used to discern atoms belonging to the Molecule itself (atom(3)=0) or the surface (atom(3)=1), which is used to apply scissors corrections or include a Kondo *U* only in the molecule.

atom(4): If atom(4)=i, the variable instructs the program to shift the on-site matrix elements of the orbitals in the atom by the voltage of Lead i. If atom(4)=0, there is no shift.

Variable atom can have very many rows and a simple MATLAB utility in 6.4 Extras 2 automates the generation of the file.

8.1.5 PARTIAL MANDATORY VARIABLE

These variables are mandatory for specific Modes only.

8.1.6 PARTIAL MANDATORY VARIABLE FOR MODE=1: VARIABLE: ERANGE

This variable sets the energy range and number of energy points used to compute;

The variable format is as follows:
# name: ERange
# type: matrix
# rows: 1
# columns: 3
ERange(1) ERange(2) ERange(3)

ERange(1) Defines *Emin* measured in eV units.
ERange(2) Defines *Emax* measured in eV units.
ERange(3) Defines *nE*.
The variable is only read ifMode=1, because the program uses default values for the other three modes: If Mode=2, *Emin* and *Emax* are set to 0 and *nE* to 1.
If Mode=3, *Emin* and *Emax* are set to ±0.0025 *Tmax* and *nE* is set to 500.
If Mode=4, *Emin* and *Emax* are set to ±*Vmax/*2, and *nE* is 300.

8.1.7 PARTIAL MANDATORY VARIABLE FOR MODE=3: VARIABLE: TRANGE.

This variable controls the temperature range in the calculation of thermal properties. It is not used in the other three modes. Its format is as follows:

# name: TRange
# type: matrix
# rows: 1
# columns: 3
TRange(1) TRange(2)TRange(3)

TRange(1) defines the minimum temperature for the calculation *Tmin*, measured in Kelvin degrees. TRange(2) defines the maximum temperature for the calculation *Tmax*, measured in Kelvin degrees. TRange(3) defines the number of temperature points to be performed in the range (*Tmin, Tmax*).

8.1.8 PARTIAL MANDATORY VARIABLE FOR MODE=4: VARIABLE: BIAS

This variable controls the calculation of current-voltage curves and is only used in Mode=4. The variable contains one row per lead. In this example for a four lead junction the first row describes how the voltage in lead 1 is to be ramped up. The voltage for the remaining leads is fixed to a given value given by *V2, V3, V4*. The format of the variable is as follows:

# name: Bias
# type: matrix
# rows: 4
# columns: 3
Bias(1) Bias(2) Bias(3)
*V*2 0 0
*V*3 0 0
*V*4 0 0

Bias(1) defines the minimum voltage to be applied to the lead *Vmin*, measured in volts.

Bias(2) defines the maximum voltage to be applied to the lead *Vmax*, measured in volts.

Bias(3) defines the number of voltage points *nvolt* to be used in the range (*Vmin, Vmax*).

8.1.9 OPTIONAL VARIABLES.

8.1.10 OPTIONAL VARIABLE FOR MODE=4: VARIABLE: BIASACCURACY

By default the energy grid for the computation of transmission and current-voltage curves contains 300 energy points which is sufficient for many junctions. However this grid might miss sharp resonances for weakly coupled junctions. In this case variable Bias_accuracy can be used to increase the energy grid. Available values are 1 and 2, whereby the energy grid is set to 1000 and 3000 energy points, respectively. Consequently the calculation is slowed down roughly by a factor of 3 or 10 respectively.

name: Bias_accuracy
type: scalar
1

8.1.11 OPTIONAL VARIABLE FOR ALL MODES: VARIABLE: EFSHIFT

G*OLLUM* uses the Fermi energy of the extended molecule to set the zero energy. This reference energy can be shifted up and down by setting EF_shift to positive or negative energy values (measured in eV).

name: EF_shift
type: scalar
1

8.1.12 OPTIONAL VARIABLE FOR ALL MODES: VARIABLE: SCISSORS

The atoms to which this correction is applied are set in variable: atom. If scissor corrections are applied then GOLLUM also prints the Molecule's Density of States (DOS) with and without scissor corrections, where thebibliographyenergy origin is chosen to be the Fermi energy of the EM region. This functionality is implemented only if the calculation is spin-unpolarized and the number of transverse k-points is one. The variable format is as follows:
# name: scissors
# type: matrix
# rows: 1
# columns: 5
scissors(1) scissors(2) scissors(3) scissors(4) scissors(5)
scissors(1) Compute Scissor corrections (if 1) or not (otherwise).
scissors(2) Number of electrons in the molecule.
scissors(3) Energy value for the shift downwards of the HOMO.
scissors(4) Energy value for the shift upwards of the LUMO.
scissors(5) Defines the distance for the screening image charge, measured in Bohr.

8.1.13 OPTIONAL VARIABLE FOR ALL MODES: VARIABLE: ANDERSON

This variable sets the parameters for the computation of Coulomb blockade and Kondo Physics in the spirit of Dynamical Mean Field Theory. The variable format is as follows:
# name: anderson
# type: matrix
# rows: 1
# columns: 3
anderson(1) anderson(2) anderson(3)

anderson(1)Compute Kondo / Coulomb blockade effects (if 1) or not (otherwise).
anderson(2)On-site Coulomb interaction U term (in eV).
anderson(3)Temperature (in Kelvin).

8.1.14 OPTIONAL VARIABLE FOR ALL MODES: VARIABLE: VGATE

This variable sets the parameter for applying a constant gate voltage to the ExtendedMolecule. The variable format is as follows:

name: Vgate
type: scalar
Vgate

VgateValue of the gate voltage (in eV).

8.1.15 EXAMPLE INPUT FILE.

The following is an input file for a two-lead junction in Mode=4.

```
 # name: Mode
# type: scalar 4
# name: Bias
# type: matrix
# rows: 1
# columns: 3
0 3.2 30
0 0 0
# name: leadp
# type: matrix
# rows: 2
# columns: 3
4 2 0
4 2 0
# name: scissors
# type: matrix
# rows:1
# columns: 5
0 42 -0.5 0.5 10000.0
 # name: atom
# type: matrix
# rows: 26
# columns: 4
1 1 4 1
…..
12 1 1 1
13 0 0 1
14 0 0 2
15 2 1 2
….
26 2 4 2
```

8.2 THE EXTENDED MOLECULE FILE.

The Extended_Molecule file enables GOLLUM to read the Hamiltonian and overlap matrices of the extended scattering region. It uses mandatory variables to define how GOLLUM reads the Fermi energy, the orbital information, the spin degrees of freedom and the number of transverse k points used in the simulation.

8.2.1 AN EXAMPLE EXTENDED_MOLECULE FILE.

```
# name: nspin
# type: scalar
1
# name: FermiE
# type: scalar
0
# name: iorb
# type: matrix
# rows: 10
# columns: 5
1 6 0 0 1
1 6 1 0 1
2 6 0 0 1
.....
5 6 1 0 1
# name: kpoints EM
# type: matrix
# rows: 1
# columns: 3
0.0000000000E+00 0.0000000000E+00 1.0000000000E+00
# name: HSM
# type: matrix
# rows: 36
# columns: 7
1 1 1 1.0 0.0 -2.0 0.0
1 3 3 1.0 0.0 -2.0 0.0
1 5 5 1.0 0.0 -2.0 0.0
...

1 10 9 0.0 0.0 -0.3 0.0
```

8.2.2 VARIABLE: NSPIN

```
# name: nspin
# type: scalar
nspin
```

This variable defines whether the calculation is spin-unpolarized (nspin=1) or polarized (nspin=2).

8.2.3 VARIABLE: FERMIE

name: FermiE
type: scalar
0
This variable defines the value of the Fermi energy at the extended molecule (in eV), which is taken as the reference energy.

8.2.4 VARIABLE: IORB

# name: iorb
# type: matrix
# rows: 10
# columns: 5
iorb(1) iorb(2) iorb(3) iorb(4) iorb(5)

This variable contains information about the orbitals used in the simulation. The variable has as many rows as orbitals exist in the EM region.

iorb(1) Indicates the atom to which the orbital belongs.
iorb(2) Indicates the principal quantum number n.
iorb(3) Indicates the orbital quantum number l.
iorb(4) Indicates lz, using real spherical harmonics.
iorb(5) Indicates the z□number.

GOLLUM only uses iorb(1). The rest are placed for information purposes.

8.2.5 VARIABLE: KPOINTS

# name: kpoints EM
# type: matrix
# rows: 1
# columns: 3
kpoints(1) kpoints(2) kpoints(3)
This variable sets the transverse kpoints used in the DFT simulation of the Extended Molecule. There are as many rows as kpoints.
kpoints(1) Gives the value of kx.
kpoints(2) Gives the value of ky.
kpoints(3) Gives the weight in the k summation.

8.2.6 VARIABLE: HSM

# name: HSM
# type: matrix
# rows: 36
# columns: 7
HSM(1) HSM(2) HSM(3) HSM(4) HSM(5) HSM(6) HSM(7) HSM(8) HSM(9)
This variables set the overlap $S(i,j,k)$ and $H(i,j,k,\uparrow\downarrow)$ for a given transverse k-point, given orbital pairs $(i,j)$and spin(↑↓). The number of columns is 7 if the calculation is paramagnetic (nspin=1) or 9 if the calculation is spin polarized (nspin=2). There are as many rows as non-zero Hamiltonian or overlap matrix elements. If for given$(i,j,k,\uparrow\downarrow)$ all matrix elements are zero, then this row is skipped.
HSM(1) Defines the k-point number as defined in variable kpoints.
HSM(2) Defines the orbital number i as defined in variable iorb.
HSM(3) Defines the orbital number j as defined in variable iorb.

HSM(4) Provides Real$[S(i,j,k)]$
HSM(5) Provides Imag$[S(i,j,k)]$
HSM(6) Provides Real$[H(i,j,k)]$ or Real$[H(i,j,k,\uparrow)]$ (nspin = 1 or 2, respectively).
HSM(7) Provides Imag$[H(i,j,k)]$ or Imag$[H(i,j,k,\uparrow)]$ (nspin = 1 or 2, respectively).
HSM(8) Exists if nspin= 2. It provides Real$[H(i,j,k,\downarrow)]$.
HSM(9) Exists if nspin = 2. It provides Imag$[H(i,j,k,\downarrow)]$..

8.3 LEAD I INPUT FILE.

8.3.1 EXAMPLE LEAD_I FILE.

There can be one of these files for each lead. The file will be read and used in the corresponding variable leadp(3) is set to ± 1. The following is an example of a Lead file:

```
# name: kpoints Lead
# type: matrix
# rows: 1
# columns: 3
0.0000000000E+00 0.0000000000E+00 1.0000000000E+00
# name: HSL
# type: matrix
# rows: 4
# columns: 11
1 1 1 1.0 0.0 -2.0 0.0 0.0 0.0 0.1 0.0
1 2 2 1.0 0.0 -1.0 0.0 0.0 0.0 0.0 0.0
1 1 2 0.0 0.0 -0.3 0.0 0.0 0.0 0.0 0.0
1 2 1 0.0 0.0 -0.3 0.0 0.0 0.0 -0.2 0.0
```

8.3.2 VARIABLE: KPOINTS_LEAD

```
# name: kpoints Lead
# type: matrix
# rows: 1
# columns: 3
kpoints(1) kpoints(2) kpoints(3)
```

This variable sets the transverse kpoints used in the DFT simulation of the Lead. There are as many rows as kpoints. It is mandatory because the ordering of k-points in the DFT simulations of the Leads and of the Extended Molecule need not coincide.
kpoints(1) Gives the value of kx.
kpoints(2) Gives the value of ky.
kpoints(3) Gives the weight in the k summation.

8.3.3 VARIABLE: HSL

```
# name: HSL
# type: matrix
# rows: 4
# columns: 11
HSL(1) HSL(2) ... HSL(11) ... HSL(15)
```

This variables set the intra- and inter-PL overlap $S_{0,1}(i,j,k)$and Hamiltonian matrix elements $H_{0,1}(i,j,k,\uparrow\downarrow)$for a given transverse k-point, given orbital pairs (i,j) and spin($\uparrow\downarrow$). The number of columns is 11 if the calculation is paramagnetic (nspin=1) or 15 if the calculation is spin-polarized (nspin=2). There are as many rows as non-zero Hamiltonian or overlap matrix elements. If for given $(i,j,k,\uparrow\downarrow)$ all matrix elements are zero, then this row is skipped. The ordering of the variable is:
For nspin = 1 (i,j,k) Real$[S_0]$ Imag$[S_0]$ Real$[H_0]$ Imag$[H_0]$ Real$[S_1]$ Imag$[S_1]$ Real$[H_1]$ Imag$[H_1]$
For nspin = 2 (i,j,k) Real$[S_0]$ Imag$[S_0]$ Real$[H_{0,\uparrow}]$ Imag$[H_{0,\uparrow}]$ Real$[H_{0,\downarrow}]$ Imag$[H_{0,\downarrow}]$ Real$[S_1]$ Imag$[S_1]$ Real$[H_{1,\uparrow}]$ Imag$[H_{1,\uparrow}$]Real$[H_{1,\downarrow}]$ Imag$[H_{1,\downarrow}]$

9 OUTPUT FILE FORMAT AND NOTATION.

The output files are written for each lead i of a junction containing N leads. Some files are universal to all Modes but some are specific to each Mode defined in the input file. The output file for calculations with N leads will have N +1 columns, where the first column is the energy (E).

9.1 THE UNIVERSAL FILES.

9.1.1 THE UNIVERSAL OUTPUT AT A GLANCE: TABLE 10.1.1.

This table shows the universal output files as seen in the Results folder in the Examples.

	Key: spin refers to paramagnetic spin up(down) refers to spin-polarized **σ** = 1 or 2	**Comment.**
Mode-1-2-3-4	Open_channels_per_spin**σ**.dat	The number of open channels per lead
Mode-1-2-3-4	Open_channels_up**σ**.dat	The number of open channels per lead
Mode-1-2-3-4	Open_channels_down**σ**.dat	The number of open channels per lead
Mode-1-2-3-4	Shot_noise_per_spin**σ**.dat	Shot noise coefficients
Mode-1-2-3-4	Shot_noise_per_up**σ**.dat	Shot noise coefficients
Mode-1-2-3-4	Shot_noise_per_down**σ**.dat	Shot noise coefficients
Mode-1-2-3-4	T_per_spin**σ**.dat	Transmission coefficients
Mode-1-2-3-4	Tup**σ**.dat	Transmission coefficients
Mode-1-2-3-4	Tdown**σ**.dat	Transmission coefficients
Mode-1-2-3-4	Bands**σ**.dat	From the GF of lead i.
Optional file initiated by scissor corrections.	DOS.dat	The EM DOS.

NOTE-1. Scissor corrections shift the HOMO and LUMO levels upwards and downwards in energy. This has the effect of reducing both the transmission and therefore the current across the junction around the Fermi energy.

NOTE-2. The results for parametric or spin polarized calculations are determined by the input Hamiltonians which for example depend upon your flags in the SIESTA input file and the value of nspin = 1 for spin-unpolarized or nspin = 2 polarized in the Extended_Molecule input file.

9.1.2 TRANSMISSION (REFLECTION) COEFFICIENTS.

Mode-1-2-3-4	T_per_spin**σ**.dat	Transmission coefficients
Mode-1-2-3-4	Tup**σ**.dat	Transmission coefficients
Mode-1-2-3-4	Tdown**σ**.dat	Transmission coefficients

Mode-1, Mode-2 and Mode-3 give zero bias calculations. For example the transmission coefficients are given by $T_{ij}(E,0)\ \ i(j) = 1 \ldots N$ in the columns where $i \neq j$ and the reflection coefficients for $i = j$. Files are written to give parametric spin calculations and up/down for spin-polarized calculations. The block of data is headed by a line stating the voltage and the first column is the energy (E) giving $N + 1$ columns in total.

Mode-4 gives finite bias results $T_{ij}(E,V)$ in the same format as the other Modes.

9.1.3 THE NUMBER OF OPEN CHANNELS.

Mode-1-2-3-4	Open_channels_per_spin**σ**.dat	The number of open channels per lead
Mode-1-2-3-4	Open_channels_up**σ**.dat	The number of open channels per lead
Mode-1-2-3-4	Open_channels_down**σ**.dat	The number of open channels per lead

These files give the number of open channels as a function of energy in the lead. The number of open channels is a dimensionless quantity.

9.1.4 SHOT NOISE.

Mode-1-2-3-4	Shot_noise_per_spin**σ**.dat	Shot noise coefficients
Mode-1-2-3-4	Shot_noise_per_up**σ**.dat	Shot noise coefficients
Mode-1-2-3-4	Shot_noise_per_down**σ**.dat	Shot noise coefficients

The file provides the Shot Noise coefficients $SN_{ij}(E, V = 0)$ using the same format as the transmission files. Shot noise coefficients are dimensionless.

9.1.5 BAND STRUCTURE.

Mode-1-2-3-4	Bands**σ**.dat	From the GF of lead i.

These files contain the inverse band structure $K(E)$ computed from the GF of lead i. Wave-vectors are measured in units of the lattice constant in the direction of transport.

9.1.6 DENSITY OF STATES (DOS).

Optional file initiated by scissor corrections flag.	DOS.dat	The EM DOS.

The file contains the DOS of the molecule with and without scissor corrections as a function of the energy.

9.2 ADDITIONAL MODE GENERATED FILES.

9.2.1 MODE GENERATED FILES AT A GLANCE: TABLE 10.2.1.

	Key: spin refers to paramagnetic spin up(down) refers to spin-polarized **i** = 1..N, where N = the number of leads.	**Comment**
Mode-2	G_spinsum**i**.dat	The spin-summed conductance G/G_0
Mode-2	G_down**i**.dat	The spin-summed conductance G/G_0
Mode-2	G_up**i**.dat	The spin-summed conductance G/G_0
Mode-2	S_matrix.dat	The S matrix
Mode-3	G_spinsum**i**.dat	The spin-summed thermal conductance $G(T)$
Mode-3	G_down**i**.dat	The spin-summed thermal conductance $G(T)$
Mode-3	G_up**i**.dat	The spin-summed thermal conductance $G(T)$
Mode-3	Thermopower**i**.dat	Spin-summed thermopower $S^e(T)$
Mode-3	Peltier**i**.dat	Spin-summed Peltier coefficients $\Pi(T)$
Mode-3	Thermal_conductance**i**.dat	Spin-summed thermal conductance $\kappa(T)$
Mode-4	Charge_Current**i**.dat	Current-voltage
Mode-4	Spin_Current**i**.dat	Spin-current

9.2.2 MODE-2 GENERATED FILES.

9.2.2.1 CONDUCTANCE.

Mode-2	G_spinsum**i**.dat	The spin-summed conductance G/G_0
Mode-2	G_down**i**.dat	The spin-summed conductance G/G_0
Mode-2	G_up**i**.dat	The spin-summed conductance G/G_0

For each lead **i** the file G_spinsum**i**.dat containing *N* columns where *N* is the number of leads is written. Each column *j* provides the spin-summed zero-voltage conductance G/G_0. For spin-polarized calculations additional files G_up**i**.dat and G_down**i**.dat containing the spin-resolved conductances G/G_0 are also given.

9.2.2.2 S MATRIX.

Mode-2	S_matrix.dat	The S matrix

The S_matrix.dat file contains the scattering matrix coefficients.

9.2.3 MODE-3 GENERATED FILES.

9.2.3.1 CONDUCTANCE.

Mode-3	G_spinsum**i**.dat	The spin-summed thermal conductance $G(T)$
Mode-3	G_down**i**.dat	The spin-summed thermal conductance $G(T)$
Mode-3	G_up**i**.dat	The spin-summed thermal conductance $G(T)$

For each lead i, GOLLUM writes a file called G_spinsumi.dat which contains the spin-summed, temperature dependent conductance ($G(T) = G/G_0$) between lead i and lead j (j≠i). There is a row for each temperature (K) where the first column in each row provides the temperature and the remaining columns give $G(T)$. The spin resolved temperature-dependent conductances are given in G_downi.dat and G_upi.dat.

9.2.3.2 THERMOPOWER.

Mode-3	Thermopower**i**.dat	Spin-summed thermopower

In a similar layout to the conductance files for each lead i GOLLUM writes a file called Thermopoweri.dat which contains the spin-summed, temperature dependent thermopower $Q(T)$ between lead i and lead j (j≠i).

9.2.3.3 PELTIER COEFFICIENTS.

Mode-3	Peltier**i**.dat	Spin-summed Peltier coefficients

In a similar layout to the conductance files for each lead i GOLLUM writes a file called Peltieri.dat which contains the spin-summed, temperature dependent Peltier coefficient between lead i and lead j (j≠i).

9.2.3.4 THERMAL CONDUCTANCE.

Mode-3	Thermal_conductancei.dat	Spin-summed thermal conductance

In a similar layout to the conductance files for each lead i GOLLUM writes a file called Thermal_conductancei.dat which contains the spin-summed, temperature dependent thermal conductance between lead i and lead j (j≠i).

9.2.4 MODE-4 GENERATED FILES.

9.2.4.1 CURRENT-VOLTAGE.

Mode-4	Charge_Currenti.dat	Current-voltage

For each lead i, GOLLUM writes a file called Charge_Currenti.dat which contains the charge current as a function of voltage $I(V)$. The file gives the total charge currents flowing between lead i and lead j (j≠i) as a function of voltage, measured in Amps.

9.2.4.2 SPIN CURRENT.

Mode-4	Spin_Currenti.dat	Spin-current

If the calculation is spin-polarized, a file called Spin Currenti.dat is also written for each lead i, containing the spin current, defined as $I\uparrow - I\downarrow$, as a function of voltage.

10 REFERENCES

1. "GOLLUM: a next-generation simulation tool for electron, thermal and spin transport" J Ferrer, C J Lambert, V M García-Suārez, D Zs Manrique, D Visontai, L Oroszlany, R Rodríguez-Ferradás, I Grace, S W D Bailey, K Guillemot, H Sadeghi, L A Algharagholy, New J. Phys. **16** 093029
2. J. M. Soler, E. Artacho, J.D. Gale, A. García, J. Junquera, P. Ordejón, and D. Sánchez-Portal, J. Phys.: Condens. Matter **14**, 2745 (2002)
3. J. P. Lewis, P. Jelnek, J. Ortega, A. A. Demkov, D. G. Trabada, B. Haycock, H. Wang, G. Adams, J. K. Tomfohr, E. Abad, H. Wang and D. A. Drabold, Phys. Stat. Solidi B **248**, 1989 (2011).

11 INDEX